



King's Research Portal

Document Version
Peer reviewed version

[Link to publication record in King's Research Portal](#)

Citation for published version (APA):

Zavatterì, M., Combi, C., Rizzi, R., & Viganò, L. (Accepted/In press). Hybrid SAT-based Consistency Checking Algorithms for Simple Temporal Networks with Decisions. In *26th International Symposium on Temporal Representation and Reasoning (TIME 2019)*

Citing this paper

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights

Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Hybrid SAT-based Consistency Checking Algorithms for Simple Temporal Networks with Decisions

Matteo Zavatteri 

Department of Computer Science, University of Verona, Italy
matteo.zavatteri@univr.it

Carlo Combi 

Department of Computer Science, University of Verona, Italy
carlo.combi@univr.it

Romeo Rizzi 

Department of Computer Science, University of Verona, Italy
romeo.rizzi@univr.it

Luca Viganò 

Department of Informatics, King's College London, UK
luca.vigano@kcl.ac.uk

Abstract

A Simple Temporal Network (STN) consists of time points modeling temporal events and constraints modeling the minimal and maximal temporal distance between them. A Simple Temporal Network with Decisions (STND) extends an STN by adding *decision time points* to model temporal plans with decisions. A decision time point is a special kind of time point that once executed allows for deciding a truth value for an associated Boolean proposition. Furthermore, STNDs label time points and constraints by conjunctions of literals saying for which *scenarios* (i.e., complete truth value assignments to the propositions) they are relevant. Thus, an STND models a family of STNs each obtained as a *projection* of the initial STND onto a *scenario*. An STND is consistent if there exists a consistent scenario (i.e., a scenario such that the corresponding STN projection is consistent). Recently, a hybrid SAT-based consistency checking algorithm (HSCC) was proposed to check the consistency of an STND. Unfortunately, that approach lacks experimental evaluation and does not allow for the synthesis of all consistent scenarios. In this paper, we propose an incremental HSCC algorithm for STNDs that (i) is faster than the previous one and (ii) allows for the synthesis of all consistent scenarios and related early execution schedules (offline temporal planning). Then, we carry out an experimental evaluation with KAPPA, a tool that we developed for STNDs. Finally, we prove that STNDs and disjunctive temporal networks (DTNs) are equivalent.

2012 ACM Subject Classification Theory of computation → Timed and hybrid models; Computing methodologies → Temporal reasoning; Computing methodologies → Planning and scheduling; Mathematics of computing → Graph algorithms; Hardware → Theorem proving and SAT solving

Keywords and phrases Simple temporal network with decisions, HSCC algorithms, incremental SAT-solving, disjunctive temporal network, KAPPA

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.2

Supplement Material <https://github.com/matteozavatteri/kappa>

1 Introduction

Context and motivations. Temporal networks are a possible formalism to model temporal plans and check the coherence of temporal constraints that impose lower and upper bounds on the temporal distance of the modeled events. A *Simple Temporal Network* (STN, [11]) is a formalism able to model an unconditional temporal plan in which all components (events



© Matteo Zavatteri and Carlo Combi and Romeo Rizzi and Luca Viganò;
licensed under Creative Commons License CC-BY

26th International Symposium on Temporal Representation and Reasoning (TIME 2019).

Editors: Johann Gamper, Sophie Pinchinat, and Guido Sciavicco; Article No. 2; pp. 2:1–2:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and their time distances) are under control. Temporal events are modeled as *time points* and their occurrence is modeled by executing the corresponding time points (i.e., by assigning them real values).

A temporal plan is *consistent* if we can schedule all the events, each at a specific time instant, such that all constraints are satisfied. If this is possible, a schedule can be synthesized for both offline planning (the plan is available before starting) or online planning (the plan is generated while executing). The difference between these two planning approaches is that in the former the consistency checking algorithm returns a solution, whereas in the latter the algorithm returns a *minimal network* to generate any solution.

However, STNs fail to model temporal plans where the occurrence of some events or the satisfaction of some constraints must happen only if some decision has been made. The decisions we are interested in have Boolean domain¹. Thus, a temporal plan subject to decisions would ask us (not) to execute some time points or to satisfy some constraints depending on what decisions we have made.

A few proposals to handle decisions within the temporal network formalisms built on top of STNs have been put forth. For instance, *Drake* [9] provides an executive for temporal plans with choices based on *Labeled STNs* that do not specify decision points (in a node-sense), whereas *Temporal Plan Networks (TPNs, [21])* extend STNs with decision nodes and model decisions as outgoing edges from such nodes. *Simple Temporal Networks with Decisions (STNDs, [4, 29])* extend STNs by adding decision points that can influence both the execution of time points and the satisfaction of constraints. Several (possibly different) STNs may arise when projecting an STND onto a scenario that models the complete interpretation of the decisions. For any scenario, we are only interested in executing the time points and satisfy the constraints *entailed* by it.

So far, only one hybrid SAT-based consistency checking algorithm (*HSCC*) has been devised to check the consistency of an STND [4]. If the algorithm finds a consistent scenario, i.e., a scenario for which the STN projection is consistent, then a solution (*scenario* plus *schedule*) exists. In this case, any schedule in the solution set involves the STN-projection corresponding to the related scenario.

This algorithm allows for an offline planning where all decisions are made before starting and the corresponding schedule is already known. However, this algorithm has never been implemented nor evaluated, and it does not allow for the synthesis of all consistent scenarios.

Contributions. Our contribution is three-fold. First, we provide **STND-HSCC2**, a novel HSCC algorithm for STNDs that (i) is faster than the existing algorithm as it rules out inconsistent scenarios as early as possible, and (ii) allows for the synthesis of all consistent scenarios and related early execution schedules (offline temporal planning). The algorithm is hybrid because a SAT-solver and a shortest path algorithm mutually influence each other (the output of the former becomes the input of the latter and vice versa continuously). Second, we discuss KAPPA, a tool that we developed for STNDs for the experimental evaluation and in which, as a minor contribution, we adapted the previous algorithm to support the synthesis of all consistent scenarios. Third, we prove that STNDs are equivalent to disjunctive temporal networks (DTNs).

¹ This is not a restriction as every finite set $\{0, 1, 2, \dots, n-1\} \subset \mathbb{N}$ of different discrete choices (i.e., decisions) can be represented in an equivalent binary notation by using $\lceil \log n \rceil$ bits where if the i^{th} bit is 1 (respectively, 0), then it means that the i^{th} decision holds (respectively, does not hold). Of course this binary representation may express more than n possibilities (e.g., to express $\{0, 1, 2\}$ we need $\lceil \log 3 \rceil = 2$ bits with which we can also express “3” when all bits are set). Again, this is not a problem if we add unsatisfiable constraints for all combinations representing numbers $\geq n$ (if any).

Organization. Section 2 provides background on STNs and STNDs, and the current consistency checking algorithm for STNDs. Section 3 discusses **STND-HSCC2**, a faster HSCC algorithm for STNDs. Section 4 discusses KAPPA and the related experimental evaluation. Section 5 proves that STNDs are equivalent to DTNs. Section 6 discusses related work. Section 7 sums up and discusses future work.

2 Background

A *Simple Temporal Network* (STN, [11]) is a pair $\langle \mathcal{T}, \mathcal{C} \rangle$, where $\mathcal{T} = \{X, \dots\}$ is a set of time points (continuous variables) and $\mathcal{C} = \{(Y - X \leq k), \dots\}$ is a set of constraints, for $X, Y \in \mathcal{T}$, $k \in \mathbb{R} \cup \pm\{\infty\}$.

An STN is *consistent* if there exists an assignment of real values to the time points such that all constraints are satisfied.

Given a consistent STN, a *schedule* is a function $S: \mathcal{T} \rightarrow \mathbb{R}$ assigning real values to time points such that if X is executed before Y , then $S(X) \leq S(Y)$. An *early execution* of an STN consists of finding a schedule executing the time points as soon as possible (e.g., by using the Floyd-Warshall algorithm [11]).

Given a set $\mathcal{P} = \{d, \dots\}$ of Boolean propositions, a *label* $\ell = \lambda_1 \dots \lambda_n$ is any finite conjunction of literals λ_i , where a literal is either d (positive literal) or $\neg d$ (negative literal), and we omit the \wedge connective to ease reading. The *empty label* is denoted by \square . The *label universe* of \mathcal{P} , denoted by \mathcal{P}^* , is the set of all possible (consistent) labels drawn from \mathcal{P} ; e.g., if $\mathcal{P} = \{d, e\}$, then $\mathcal{P}^* = \{\square, d, e, \neg d, \neg e, de, d\neg e, \neg de, \neg d\neg e\}$. Two labels ℓ_1, ℓ_2 are *consistent* if their conjunction $\ell_1 \ell_2$ is satisfiable. A label ℓ_1 *entails* a label ℓ_2 (written $\ell_1 \Rightarrow \ell_2$) if all literals in ℓ_2 appear in ℓ_1 too (i.e., if ℓ_1 is more *specific* than ℓ_2). For instance, if $\ell_1 = d\neg e$ and $\ell_2 = d$, then ℓ_1 and ℓ_2 are consistent since $d\neg ed$ is satisfiable, and ℓ_1 entails ℓ_2 since $d\neg e \Rightarrow d$.

A *scenario* is a mapping $s: \mathcal{P} \rightarrow \{\top, \perp\}$ assigning a truth value to each $d \in \mathcal{P}$. A scenario *satisfies* a label ℓ (in symbols $s \models \ell$) if ℓ evaluates to true under the interpretation given by s (e.g., if $s(d) = \top$ and $s(e) = \perp$, then $s \models d\neg e$).

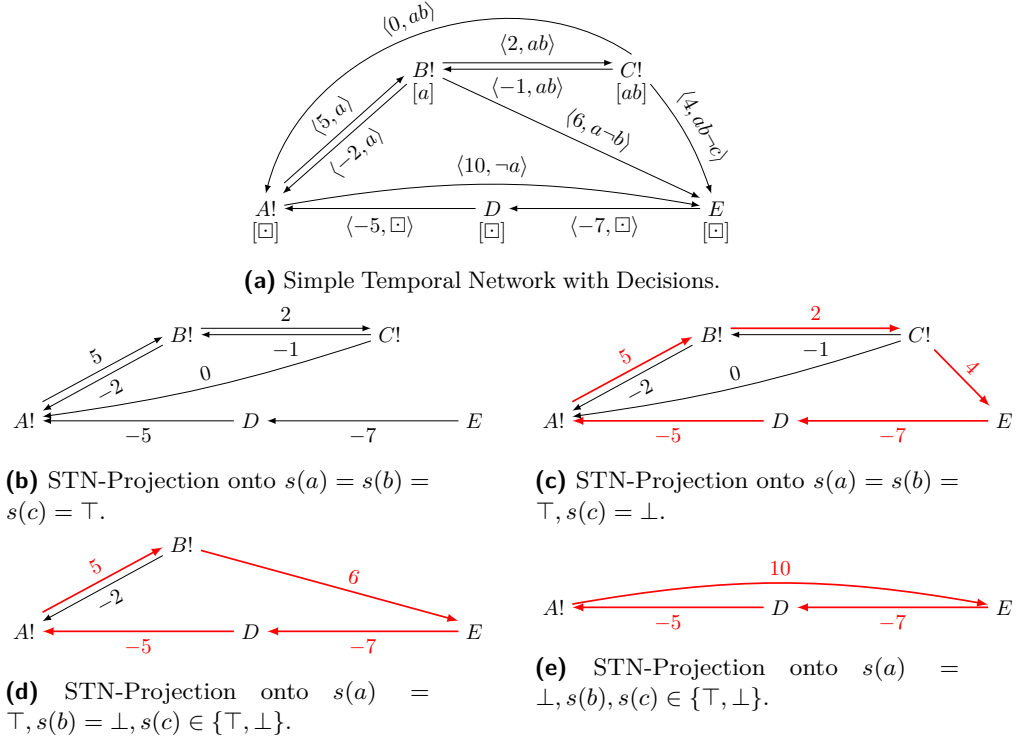
► **Definition 1.** A Simple Temporal Network with Decisions (STND, [4, 29, 33]) is a tuple $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$, where:

- $\mathcal{T} = \{X, \dots, Z\}$ is a finite set of time points.
- $\mathcal{DT} \subseteq \mathcal{T} = \{D!, \dots, H!\}$ is a finite set of decision time points.
- $\mathcal{P} = \{d, \dots, g\}$ is a finite set of Boolean propositions.
- $O: \mathcal{P} \rightarrow \mathcal{DT}$ is a bijection assigning a unique proposition to each decision time point $D!$ that controls the truth value assignment to d ($O^{-1}: \mathcal{DT} \rightarrow \mathcal{P}$ models the inverse).
- $L: \mathcal{T} \rightarrow \mathcal{P}^*$ is a function assigning labels to time points.
- \mathcal{C} is a finite set of labeled constraints $(Y - X \leq k, \ell)$ where $X, Y \in \mathcal{T}$, $k \in \mathbb{R} \cup \pm\{\infty\}$ and $\ell \in \mathcal{P}^*$.

The *STN-projection* of an STND \mathcal{S} with respect to a scenario s (written $\pi_s(\mathcal{S})$) is an STN $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ built as follows:

- $\mathcal{T}_s = \{X \mid X \in \mathcal{T} \wedge s \models L(X)\}$
- $\mathcal{C}_s = \{(Y - X \leq k) \mid (Y - X \leq k, \ell) \in \mathcal{C} \wedge s \models \ell\}$

\mathcal{S} is *consistent* if there exists a scenario s such that $\pi_s(\mathcal{S})$ is consistent. A *solution* is a pair $\langle s, S \rangle$, where s is a scenario, S is a schedule with domain \mathcal{T}_s , and $S(Y) - S(X) \leq k$ holds for each $(Y - X \leq k) \in \mathcal{C}_s$. Checking consistency of STNDs is NP-complete [4].



■ **Figure 1** An example of STND and related STN-projections (where thick red edges highlight negative cycles).

We represent an STND graphically by extending the distance graph of an STN into a labeled distance (multi)graph. The set of nodes still coincides with the set of time points, whereas each edge $X \rightarrow Y$ labeled by $\langle k, \ell \rangle$ represents $(Y - X \leq k, \ell) \in \mathcal{C}$. Time points' labels are shown below the nodes. Many $\langle k, \ell \rangle$ can be specified for the same $X \rightarrow Y$ provided their ℓ are different (if two labels are equal, we keep the smallest k). Figure 1a shows an example of STNDs, whereas Figures 1b–1e show its STN-projections.

A label ℓ labeling a time point or a constraint is *honest* if for each literal d or $\neg d$ in ℓ we have that $\ell \Rightarrow L(D!)$, where $D! = O(d)$ is the decision time point associated to d ; ℓ is dishonest otherwise. For example, consider $L(C!) = ab$ in Figure 1a. $C!$ appears in a solution only if $s(a) = s(b) = \top$. However, deciding \top for b implies that $B!$ appears in the solution too, which in turn requires that $A!$ appears (before $B!$) in the same solution with \top decided for a . Thus, an honest ℓ containing b or $\neg b$ should also contain a . A label on a constraint is *coherent* if it is at least as expressive as the labels of the time points appearing in the constraint (i.e., ℓ contains all literals in the labels of the two connected time points).

► **Definition 2** (Well-definedness). *An STND is well-defined [17, 29, 33] if*

- $L(X) \Rightarrow L(O(d))$ and $(O(d) - X \leq 0) \in \mathcal{C}$ for each $X \in \mathcal{T}$ and $\{d, \neg d\} \in L(X)$, and
- $\ell \Rightarrow L(Y) \wedge L(X)$ for each $(Y - X \leq k, \ell) \in \mathcal{C}$, and $\ell \Rightarrow L(O(d))$ for each literal $\{d, \neg d\} \in \ell$.

Figure 1a is an example of well-defined STND modeling a temporal plan with 3 decisions ($A!$, $B!$ and $C!$) and two (instantaneous) activities (D and E). $A!$ is the first time point to execute. We execute $B!$ if and only if we decided \top for a ($L(B!) = a$) and $C!$ if we

Algorithm 1: STND-HSCC1(\mathcal{S})**Input:** An STND $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, C \rangle$.**Output:** A solution $\langle s, S \rangle$ for \mathcal{S} if \mathcal{S} is consistent; “*inconsistent*” if \mathcal{S} is inconsistent.

STN-CC is a consistency checking algorithm for STNs.

```

1  $\varphi \leftarrow \bigwedge_{p \in \mathcal{P}} (p \vee \neg p)$  ▷ Make every assignment possible
2 while true do
3    $s \leftarrow \text{SAT-SOLVE}(\varphi)$  ▷ Try to find a satisfying assignment  $s$ 
4   if  $\varphi$  is unsatisfiable then
5     return inconsistent
6    $\langle \mathcal{T}_s, \mathcal{C}_s \rangle \leftarrow \pi_s(\mathcal{S})$  ▷ Project  $\mathcal{S}$  onto  $s$ 
7   if  $\text{STN-CC}(\langle \mathcal{T}_s, \mathcal{C}_s \rangle)$  then
8     return  $\langle s, S \rangle$  ▷ where  $S$  is an early schedule for  $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ 
9    $\varphi \leftarrow \varphi \wedge \text{CUT-SCENARIO}(\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle)$  ▷ Exclude this scenario

```

152 further decided \top for b too ($L(C!) = ab$). Instead, $A!$, D and E are always executed
 153 ($L(A!) = L(D) = L(C) = \square$). We can execute $B!$ (if we decide so) after at least 2 ($B! \rightarrow A!$
 154 labeled by $\langle -2, a \rangle$) and within 5 time units since $A!$ was executed ($A! \rightarrow B!$ labeled by
 155 $\langle 5, a \rangle$). The same happens for $C!$ with respect to $B!$ (after 1 and within 2 time units since
 156 $B!$). Instead, we always execute D after minimum 5 time units since $A!$ ($D \rightarrow A!$ labeled by
 157 $\langle -5, \square \rangle$) and E after 7 time units since D ($E \rightarrow D$ labeled by $\langle -7, \square \rangle$). Furthermore, if we
 158 decide

- 159 ■ \perp for a , then we must execute E within 10 time units since $A!$ ($A! \rightarrow E$ labeled by
 160 $\langle 10, \neg a \rangle$),
- 161 ■ \top for a and \perp for b , then we must execute E within 6 time units since $B!$ ($B! \rightarrow E$
 162 labeled by $\langle 6, a \neg b \rangle$),
- 163 ■ \top for both a and b and \perp for c , then we must execute E within 4 time units since $C!$
 164 ($C! \rightarrow E$ labeled by $\langle 4, ab \neg c \rangle$).

165 Therefore negative values on edges model delays, positive ones model deadlines.

166 To check the consistency of an STND, we can use the *hybrid SAT-based consistency*
 167 *checking (HSCC)* algorithm proposed in [4]. STND-HSCC1, specified in Algorithm 1, (STND-CC
 168 in [4]) maintains a formula φ specifying CNF clauses over propositions in \mathcal{P} . Initially, φ
 169 allows for all truth value assignments. In each round of the algorithm we ask the SAT
 170 solver for a truth value assignment making φ true. Such an assignment (if any) corresponds
 171 to a scenario s over which we can project the STND and check if the resulting STN is
 172 consistent (“SAT-solver influences directed weighted graph algorithm”). If so, we return this
 173 scenario and a valid schedule for the projected STN (i.e., a solution). Otherwise, we apply
 174 De Morgan’s rules to the negation of the *relevant part* of the scenario containing the negative
 175 cycle (CUT-SCENARIO in Algorithm 2) and add the resulting clause to φ and go ahead with
 176 the next round (“directed weighted graph algorithm influences the SAT-solver”). This makes
 177 the approach hybrid. If φ has become unsatisfiable it means that all STN-projections are
 178 inconsistent and therefore the STND is inconsistent. Similar approaches are described in
 179 [23, 28].

180 An example of round for the network in Figure 1a is as follows. Suppose that $\text{SAT-SOLVE}(\varphi) =$
 181 $ab \neg c$ (i.e., $s(a) = s(b) = \top$ and $s(c) = \perp$). Since the STN $\pi_{ab \neg c}(\mathcal{S})$ is inconsistent (Figure 1c
 182 admits a negative cycle), we add to φ the clause $\neg(a \wedge b \wedge \neg c)$, which simplifies to $(\neg a \vee \neg b \vee c)$,
 183 to ask the SAT solver for a different truth value assignment excluding this projection (if
 184 any). Figure 1a is consistent if and only if $s(a) = s(b) = s(c) = \top$. A possible schedule for

Algorithm 2: CUT-SCENARIO($\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle$)

Input: An STND $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$ and one of its STN-projections $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$
Output: A clause ψ expressing the cut of the relevant part of the scenario.

```

1  $\psi \leftarrow \top$   $\triangleright \psi$  will contain the relevant part of  $s$ 
2 foreach constraint  $c \in \mathcal{C}_s$  do
3    $\psi \leftarrow \psi \wedge \ell_c$   $\triangleright$  where  $\ell_c$  is the corresponding label of  $c$  in  $\mathcal{S}$ 
4 return DeMorgan( $\neg(\psi)$ )  $\triangleright$  the clause expressing the cut of  $\psi$ 

```

185 Figure 1b is $S(A!) = 0, S(B!) = 2, S(C!) = 3, S(D) = 5, S(E) = 12$. Any other combination
 186 leads to a projection containing a negative cycle (Figures 1c-1e).

3 A Faster HSCC Algorithm for STNDs

188 STND-HSCC1 is correct [4], but it suffers from the limitation that projections are tested only
 189 when the SAT solver returns a complete truth value assignment. Consider Figure 1a and
 190 assume that the SAT solver starts on the formula $\varphi = (a \vee \neg a) \wedge (b \vee \neg b) \wedge (c \vee \neg c)$, which makes
 191 every truth value assignment possible. Suppose that in the search tree the SAT solver decides
 192 \perp for a proposition d going down to the left and \top going down to the right in the search
 193 tree, and assume that the order of visit is left then right. The first truth value assignment
 194 returned is $a = \perp, b = \perp$ and $c = \perp$ (corresponding to the scenario $s(a) = s(b) = s(c) = \perp$).
 195 Now STND-HSCC1 would project the STND in Figure 1a onto s to obtain the STN shown
 196 in Figure 1e and eventually detect the negative cycle. However, the negative cycle could
 197 have been detected much earlier, say, when a was assigned \perp . Indeed, all projections of
 198 any scenario containing $s(a) = \perp$ boil down to Figure 1e (no matter which Boolean values
 199 are assigned to b and c). Therefore, a clever implementation of this algorithm calls for an
 200 early detection of negative cycles. Before proceeding with it, we must refine the concept of
 201 scenario and projection so that they support “unknown” propositions (i.e., propositions that
 202 have not been assigned a value yet).

203 ► **Definition 3.** A scenario is (now) a mapping $s: \mathcal{P} \rightarrow \{\top, \perp, -\}$ assigning either true,
 204 false or unknown to each proposition $d \in \mathcal{P}$. A scenario s satisfies a label ℓ if ℓ evaluates to
 205 true under the following interpretation given by s :

- 206 1. $s \models \lambda$ iff $(\lambda = d \wedge s(d) = \top)$ or $(\lambda = \neg d \wedge s(d) = \perp)$,
- 207 2. $s \models \ell$ iff $s \models \lambda_1$ and ... and $s \models \lambda_n$ for $\ell = \lambda_1 \dots \lambda_n$.

208 Note that s never satisfies a label containing a literal for which the corresponding proposition
 209 is unknown in s .

210 The definition of STN projection remains the same as that given in Definition 1 but
 211 extended to the new definition of scenario. As a result, Figure 1e now becomes a representative
 212 also for any scenario s such that $s(a) = \perp$ and $s(b), s(c) \in \{\top, \perp, -\}$. Another example is
 213 Figure 1d, extending $s(c) \in \{\top, \perp\}$ to $s(c) \in \{\top, \perp, -\}$. Now we have everything we need to
 214 hunt down inconsistent scenarios as early as possible.

215 STND-HSCC2 (Algorithm 3) is a brand new algorithm to check the consistency of STNDs.
 216 It allows for the synthesis of a single or all scenarios admitting a consistent schedule for
 217 the corresponding STN-projection. STND-HSCC2 still initializes a CNF formula φ making all
 218 truth value assignments possible. Then, it starts the SAT-solver and hooks a listener to the
 219 corresponding run. Such a listener is able to operate on φ by adding CNF clauses on the fly
 220 if needed and is triggered by two main events: *assume* and *solution found*.

Algorithm 3: STND-HSCC2(\mathcal{S}, all)

Input: An STND $\mathcal{S} = \langle \mathcal{T}, \mathcal{DT}, \mathcal{P}, O, L, \mathcal{C} \rangle$ and a Boolean value *all* meaning *all consistent scenarios* iff $all = \top$.

Output: A single or all scenarios s along with schedule(s) S for the projection $\pi_s(\mathcal{S})$ if \mathcal{S} is consistent, “*inconsistent*” otherwise.

```

1  $\varphi \leftarrow \bigwedge_{p \in \mathcal{P}} (p \vee \neg p)$  ▷ Make every assignment possible
2  $Sols = \emptyset$  ▷ The set of (all) consistent scenarios (global variable)
3 Hook a listener to the run of the SAT solver and make it detect negative cycles as early as possible (on blocks (below) define the event-driven behavior).
4 SAT-SOLVE( $\varphi$ )
5 if  $Sols = \emptyset$  then
6   return inconsistent
7 return  $Sols$ 
8 on assume  $d = \top$  or assume  $d = \perp$ : ▷ Partial model
9   Build a scenario  $s$  from the current truth value assignment extended with  $s(d) = \top$  or  $s(d) = \perp$  (depending on the case)
10   $\langle \mathcal{T}_s, \mathcal{C}_s \rangle \leftarrow \pi_s(\mathcal{S})$  ▷ Get the STN projection
11  if BellmanFord( $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ ) detects a negative cycle then
12     $\varphi \leftarrow \varphi \wedge \text{CUT-SCENARIO}(\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle)$  ▷ Add clause
13 on solution found: ▷ Complete model
14   Build a scenario  $s$  from the current truth value assignment
15    $\langle \mathcal{T}_s, \mathcal{C}_s \rangle \leftarrow \pi_s(\mathcal{S})$  ▷ Get the STN projection
16   if BellmanFord( $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ ) does not detect any negative cycle then
17      $Sols \leftarrow Sols \cup \{ \langle s, S \rangle \}$  ▷  $S$  is an early schedule for  $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ 
18   if BellmanFord( $\langle \mathcal{T}_s, \mathcal{C}_s \rangle$ ) detects negative cycle or all is true then
19      $\varphi \leftarrow \varphi \wedge \text{CUT-SCENARIO}(\mathcal{S}, \langle \mathcal{T}_s, \mathcal{C}_s \rangle)$  ▷ Add clause

```

221 An *assume* ($d = \top$ or $d = \perp$) event (Algorithm 3, lines 8-12) triggers an action of the
 222 listener to “look ahead” if the STN-projection obtained by projecting the STND onto the
 223 scenario built from the current truth value assignment and *extended* with this assumption
 224 contains a negative cycle. If so, we extend φ by adding (on the fly) a CNF clause modeling
 225 the negation of the part of s containing a negative cycle in order to avoid getting the same
 226 scenario again. If the projection is consistent, STND-HSCC2 does nothing and lets the run go.

227 A *solution found* event (Algorithm 3, lines 13-19) extends the behavior of the listener
 228 described for *assume* as follows. When triggered, the listener builds a scenario from the
 229 current truth value assignment (which does not need to be extended with anything else this
 230 time). Then, it checks if the corresponding STN-projection contains a negative cycle. If it
 231 does not, then it computes an (early) schedule S for the STN projected onto s and adds the
 232 pair $\langle s, S \rangle$ to the set of solutions. If it detects a negative cycle (or all consistent scenarios
 233 are sought), then it acts as for *assume* events.

234 Eventually, when the run of the SAT solver ends, either $Sols = \emptyset$ (and thus the starting
 235 STND is inconsistent), or $Sols$ contains at least 1 solution (scenario-schedule).

236 Like STND-HSCC1, STND-HSCC2 is sound and complete because it is based on a SAT-solver
 237 that allows us to iterate on all the models. Whenever we add a clause, we exclude a relevant
 238 part of a scenario that we do not want to get anymore. The sooner, the better.

239 Besides the SAT solver, all other internal sub-procedures (mostly, algorithms for directed
 240 weighted graphs) are well known to be sound and complete, and run in polynomial time.

4 Experimental Evaluation

We developed KAPPA, a tool for STNDs that takes in input a specification of an STND and acts both as a solver and as a solution verifier. KAPPA relies on SAT4J [3], a Java library compliant with the IPASIR interface that specifies how to interact with a SAT solver [1].

KAPPA implements both STND-HSCC1 and STND-HSCC2. We extended STND-HSCC1 so that it allows for the synthesis of all consistent scenarios as well. In this way, we could carry out a more accurate experimental evaluation comparing the two algorithms when seeking single or all consistent scenarios.

We randomly generated 2200 STNDs partitioned in benchmark 11 sets, each one containing 100 consistent STNDs and 100 inconsistent STNDs. Regardless of the set, each STND has exactly 100 time points. The first set (100TimePoints/10Decisions) specifies 10 decision time points, the second set (100TimePoints/11Decisions) specifies 11 decision time points and so on, up to the eleventh one (100TimePoints/20Decisions) that specifies 20 decision time points. Each STND has a maximum number of constraints of $|\mathcal{T}| \times |\mathcal{DT}|$. Time points and constraints are randomly labeled so that the resulting STND is well defined. The weights on labeled edges range from -100 to 100. See the link “Supplement Material” before Section 1 to get KAPPA and these benchmark sets.

We ran KAPPA on these benchmark sets without imposing any limit to collect data (time and space) for both STND-HSCC1 and STND-HSCC2 when seeking a single or all consistent scenarios.

We graphically show the results in Figure 2, where x-axes always represent the number (#) of decision time points (i.e., the set under analysis) and y-axes represent either the average time elapsed or space consumed when analyzing the instances in that set.

Figure 2a shows the results of the analysis run on the sets containing consistent STNDs when seeking a single consistent scenario. The graph shows that STND-HSCC2 is significantly faster than STND-HSCC1 for STNDs specifying more than 16 decision time points. Figure 2b shows the results of the same analysis when seeking all consistent scenarios: despite a normal general worsening of performances (all consistent scenarios are sought and not just one) STND-HSCC2 is faster than STND-HSCC1 for STNDs specifying 20 decisions. Figure 2c shows the results of the analysis on the sets containing inconsistent STNDs. STND-HSCC2 has no competitors here, whereas STND-HSCC1 starts having a serious exponential blow up for STNDs specifying more than 14 decisions. Figure 2d shows the average space consumed when synthesizing all consistent scenarios. The curve grows exponentially according to the number of decision time points (recall that STND-HSCC1 and STND-HSCC2 return the same set of consistent scenarios in such an analysis, therefore we only show the data for STND-HSCC2).

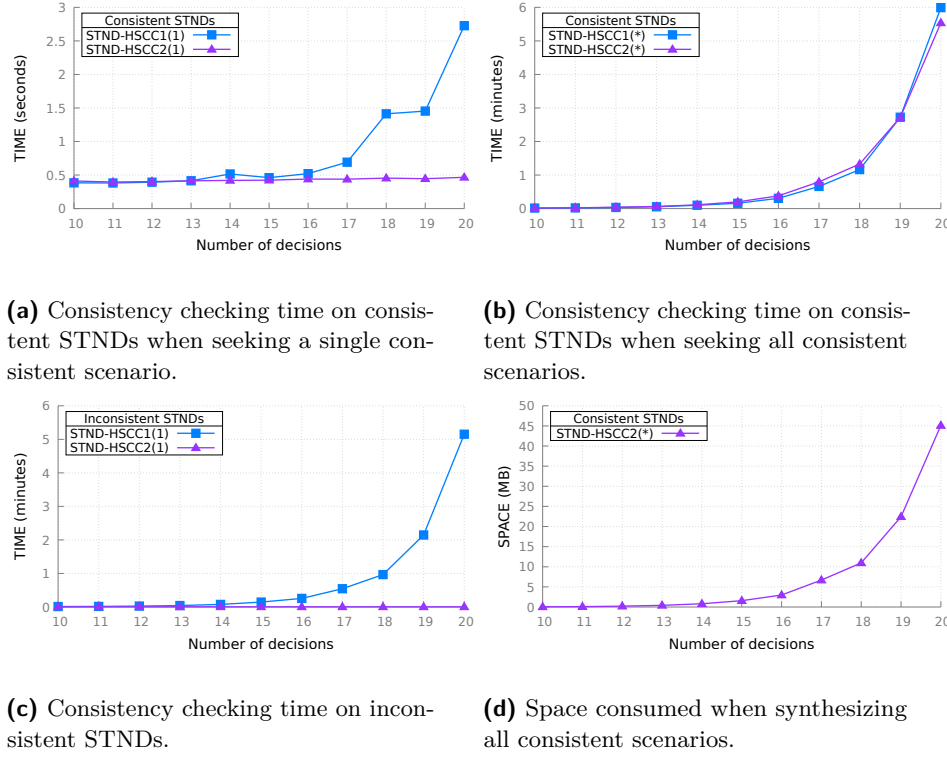
We verified all synthesized solutions. No constraint was violated.

5 Equivalence with Disjunctive Temporal Networks

Disjunctive temporal networks (DTNs, [25]) allow for disjunctions of temporal constraints (i.e., alternatives) in a temporal problem and are a possible formalism to model the *disjunctive temporal problem* (DTP). For example, we might want that once an event modeled by a time point X happened, another event modeled by a time point Y happens *either* after 10 (seconds, minutes, hours, ...) *or* within 5. Such a constraint would look like

$$(X - Y \leq -10) \vee (Y - X \leq 5)$$

Any assignment of real values to X and Y satisfies the constraint if it satisfies (at least) one disjunct.



■ **Figure 2** Experimental evaluation with KAPPA.

286 Differently from the initial proposal in [11], where disjunctions of intervals were allowed
 287 on the *same pairs of time points* only, the work we consider here is the one in [25], not having
 288 such a restriction.

289 ► **Definition 4.** A disjunctive temporal network (DTN) is a pair $\langle \mathcal{T}, \mathcal{C} \rangle$, where

290 ■ \mathcal{T} is the usual finite set of time points, and

291 ■ \mathcal{C} is a finite set of temporal constraints each one having the form

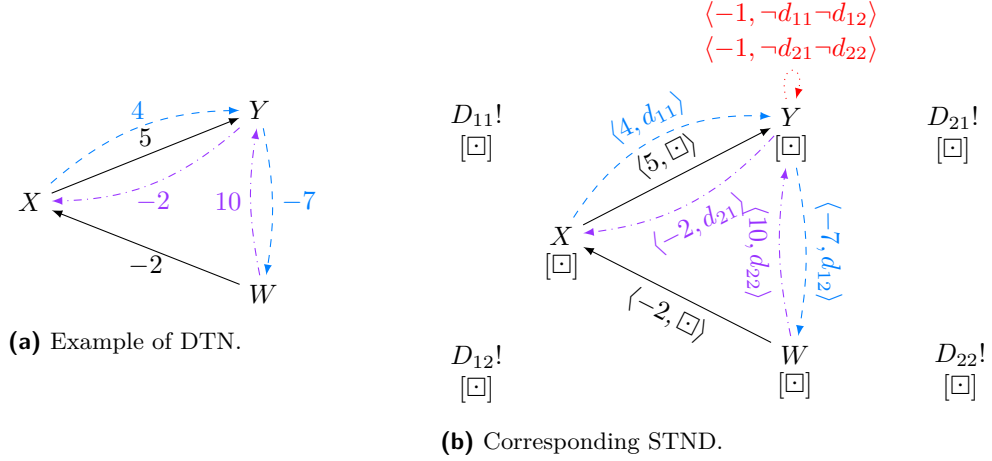
$$292 \underbrace{(Y_1 - X_1 \leq k_1) \vee \dots \vee (Y_n - X_n \leq k_n)}_{n \text{ disjuncts (atoms)}}$$

293 where $X_i, Y_i \in \mathcal{T}$ and $k_i \in \mathbb{R}$. A temporal constraint is non-disjunctive if and only if it
 294 contains one disjunct, disjunctive otherwise.

295 A DTN is consistent if there exists an assignment of real values to all time points (i)
 296 always satisfying all non-disjunctive constraints and (ii) satisfying at least one disjunct
 297 for each disjunctive constraint.

298 We write $D(i)$ to shorten the i^{th} disjunctive constraint and more specifically $D(i, j)$ to
 299 refer to the j^{th} disjunct of the i^{th} disjunctive constraint [25].

300 We represent a DTN graphically through a *colored multi graph*, where *black edges* model
 301 non-disjunctive constraints (i.e., those constraints that must always hold), whereas *colored*
 302 *edges* (different from black) model disjunctive constraints (i.e., those $D(i)$ s for which at least
 303 one disjunct must hold). Each disjunctive constraint is assigned to a different color (we also
 304 use a unique line pattern for each color).



■ **Figure 3** Representing and encoding DTNs into STNDs.

To give an example, consider the following DTN, whose corresponding colored multi graph is shown in Figure 3a.

■ $\mathcal{T} = \{X, Y, W\}$

$$\mathcal{C} = \left\{ \overbrace{(Y - X \leq 5), (X - W \leq -2)}^{\text{must always hold}}, \underbrace{(Y - X \leq 4) \vee (W - Y \leq -7)}_{D(1)}, \underbrace{(X - Y \leq -2) \vee (Y - W \leq 10)}_{D(2)} \right\}$$

$\underbrace{\quad\quad\quad}_{D(2,1)} \quad \underbrace{\quad\quad\quad}_{D(2,2)}$

The DTN in Figure 3a is consistent, and, for example, the assignment $X = 0$, $Y = 3$ and $W = 5$ satisfies:

- all non-disjunctive constraints (solid black edges),
- $D(1, 1)$ but not $D(1, 2)$ for the disjunctive constraints $D(1)$ (dashed blue edges),
- $D(2, 1)$ and also $D(2, 2)$ for the disjunctive constraint $D(2)$ (dashdotted purple edges).

We now proceed by proving that STNDs and DTNs are equivalent. We first give a strongly polynomial time encoding from DTNs to STNDs and then the vice versa (and we provide examples throughout this discussion).

5.1 Encoding DTNs into STNDs

We encode the DTN in Figure 3a into the corresponding STND in Figure 3b as follows.

We generate a “core” STND containing all time points and all non-disjunctive constraints of the starting DTN and labeling them by \Box , since all time points must always be assigned a value and all non-disjunctive constraints must always be satisfied.

For each disjunctive constraint $D(i)$ in the DTN, we add to the STND as many decision time points $D_{ij}!$ as the number of disjuncts $D(i, j)$. These decision time points are not constrained to any other time point in the STND (i.e., free to take any value). Any disjunct $D(i, j)$ in the DTN appears as a constraint in the STND labeled by d_{ij} (the proposition associated to $D_{ij}!$) so that when $d_{ij} = \top$, the disjunct of the DTN (labeled constraint in the STND) must hold and when $d_{ij} = \perp$ we are not obliged to satisfy it. Moreover, we impose

that at least one disjunct $D(i, j)$ for any disjunctive constraint $D(i)$ must hold (otherwise, it would be possible to disable them all by setting all d_{ij} to \perp). We enforce this condition by adding a negative self loop labeled by $\neg d_{i,j_1} \dots \neg d_{i,j_n}$ on any time point of the STND.

In Figure 3b we added four decision time points $D_{11}!$, $D_{12}!$, $D_{21}!$ and $D_{22}!$ and the labeled constraints $X \rightarrow Y$ labeled by $\langle 4, d_{11} \rangle$, $Y \rightarrow W$ labeled by $\langle -7, d_{12} \rangle$, $Y \rightarrow X$ labeled by $\langle -2, d_{21} \rangle$ and $W \rightarrow Y$ labeled by $\langle 10, d_{22} \rangle$ to switch on and off the disjuncts $D(1, 1)$, $D(1, 2)$, $D(2, 1)$ and $D(2, 2)$ through the truth value assignments to d_{11} , d_{12} , d_{21} and d_{22} . Finally, we added two negative self loops $Y \rightarrow Y$ labeled by $\langle -1, \neg d_{11} \neg d_{12} \rangle$ and $\langle -1, \neg d_{21} \neg d_{22} \rangle$ to prevent a disjunctive constraint $D(i)$ from being excluded (red self loop at Y). Note that the “-1” is meaningless: *any negative number* (e.g., -3, -159 or $-\epsilon$) is fine for this purpose. Likewise, the choice of time point Y is meaningless too. Any time point would be fine for this purpose (e.g., $X \rightarrow X$ labeled by the same constraints). Negative self loops are the more intuitive way to enforce these conditions. However, nothing would have prevented us from creating cycles of negative sum with respect to these labels involving many time points.

To ease reading and understand “what goes where”, we colored the STND in Figure 3b with the same colors of the DTN in Figure 3a and showed the added negative cycles in red.

This encoding is strongly polynomial. The number of time points in the STND is equal to the number of time points in the DTN plus as many decision time points as the number of disjuncts $D(i, j)$ contained in all disjunctive constraints $D(i)$ in the DTN. The number of constraints in the STND is equal to the the number of non-disjunctive constraints plus as many constraints as the number of disjuncts $D(i, j)$ contained in all disjunctive constraints $D(i)$ in the DTN plus as many constraints as the number of disjunctive constraints $D(i)$ to model negative loops.

Any consistent scenario in the STND says which disjuncts (at least one for each disjunctive constraint) are satisfied for the solution. If the STND is inconsistent, then so is the DTN.

5.2 Encoding STNDs into DTNs

We encode the STND in Figure 4a into the corresponding STND in Figure 4b as follows.

First of all, if the STND has labels on nodes we convert it to its streamlined version having only label on edges. The process of streamlining a temporal network was first discussed in [5] for CSTNs. However, that process works for STNDs as well (as consistency is basically entailed by controllability). Then, we generate a “core” DTN having the same set of time points of the STND (we drop all “!” from the names) and all constraints labeled by \square in the STND as non-disjunctive constraints in the DTN.

For each proposition d associated to a decision time point $D!$ in the STND, we add to the DTN a time point d and the disjunctive constraint

$$\underbrace{(d - D \leq 0)}_{\text{means } d = \perp} \vee \underbrace{(D - d \leq -1)}_{\text{means } d = \top}$$

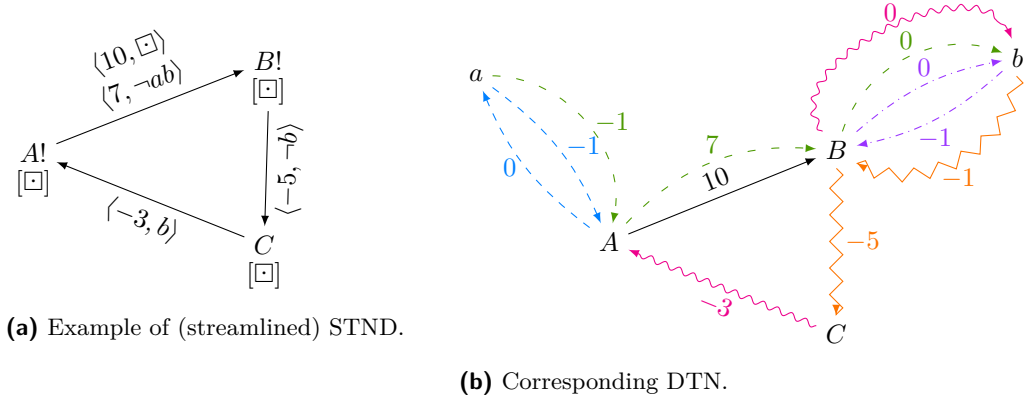
where the former says that d “occurs within” D , whereas the latter says that d occurs at least 1 after D (a way to to simulate a Boolean condition).

Now, every constraint $X \rightarrow Y$ labeled by $\langle k, d \neg e f \dots \rangle$ (in the STND) implies the following “meta constraint” in the DTN:

$$\underbrace{(D - d \leq -1)}_d \wedge \underbrace{e - E \leq 0}_{\neg e} \wedge \underbrace{F - f \leq -1}_{f} \dots \Rightarrow Y - X \leq k$$

which can be rewritten as

$$\neg(D - d \leq -1 \wedge e - E \leq 0 \wedge F - f \leq -1 \dots) \vee Y - X \leq k$$



■ **Figure 4** Encoding STNDs into DTNs.

and finally simplified to

$$\underbrace{(d - D \leq 0)}_{\neg d} \vee \underbrace{(E - e \leq -1)}_e \vee \underbrace{(f - F \leq 0)}_{\neg f} \cdots \vee (Y - X \leq k)$$

Note that for any D and d (in the DTN) we define $\neg(D - d \leq -1)$ as $d - D \leq 0$ and $\neg(d - D \leq 0)$ as $D - d \leq -1$ since they are abstracting Boolean conditions only and we are not therefore interested in a specific numeric value. Therefore, for any labeled constraint in the STND we add such a disjunctive constraint to the DTN.

In Figure 4b we add two time points a and b and the following constraints:

- $B - A \leq 10$ (solid black edges),
- $D(1): (a - A \leq 0) \vee (A - a \leq -1)$ (dashed blue edges),
- $D(2): (b - B \leq 0) \vee (B - b \leq -1)$ (dashdotted purple edges),
- $D(3): (A - a \leq -1) \vee (b - B \leq 0) \vee (B - A \leq 7)$ (loosely dashed green edges),
- $D(4): (B - b \leq -1) \vee (C - B \leq -5)$ (zigzag orange edges),
- $D(5): (b - B \leq 0) \vee (A - C \leq -3)$ (snake magenta edges).

We show the “colored” DTN graph in Figure 4b. Now, the DTN is consistent if and only if the STND is so. A solution of the DTN corresponds to a consistent scenario in the STND. The truth value assignment to the propositions in the STND depends on the real value assignments to the time points modeling those propositions in the DTN. For any proposition d in the STND, d is false iff in the DTN the time point d has a value not greater than D ! and d is true in the STND iff in the DTN the value of time point d is greater than D ! (the assignment to the other time points defines a schedule consistent for the scenario).

This encoding is strongly polynomial. The number of time points in the DTN is the same of that in the STND plus as many time points as the number of propositions in the STND. The number of constraints in the DTN is given by the number of unlabeled constraints in the STND (non-disjunctive constraints in the DTN), plus as many disjunctive constraints as the number of labeled constraints in the STND (whose labels are different from \square). Also, for any disjunctive constraint $D(i)$ in the DTN, the number of disjuncts of $D(i)$ is $n + 1$ where n is the number of literals contained in the label of the corresponding constraint in the STND and the “+1” refers to the inequality.

6 Related Work

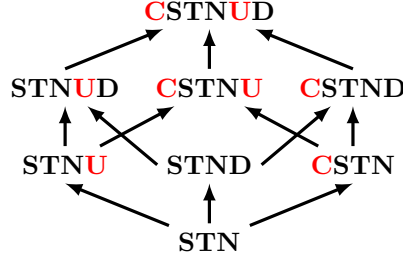
STNs [11] model fully-controllable plans but do not employ decisions. *Drake* [9] is an executive for temporal plans with choices modeled as Labeled STNs that extend STNs by labeling constraints with environments (set of instantiated discrete variables). There are no decision points and time points are unlabeled. During execution, choices are discriminated by generating conflicts according to the timing Drake decides to schedule some event. STNDs differ from Drake in their specification and in how decisions are made. STNDs rely on a more structured approach by using labels instead of environments, labeling time points to prevent them from being executed when some literal in the label is still unknown, enforcing well-defined properties and making decisions only upon the execution of the related decision time points. A *Disjunctive Temporal Network (DTN)* [25] extends an STN with disjunctive constraints. Any solution to a DTN must satisfy all non-disjunctive constraints (i.e., STN-constraints) and at least one disjunct for each disjunctive constraint. Labeled STNs and DTNs are equivalent [9]. DTNs and STNDs are equivalent too, therefore, Labeled STNs are equivalent to STNDs as well.

Temporal Plan Networks (TPNs) [21] extend STNs by adding decision nodes and symbolic constraints to model temporal plans with controllable choices modeled as outgoing edges from a decision node. Taking one of these outgoing edges means making a particular decision. Time points are not labeled and activities are modeled as pair of non-decision nodes (start,end). A symbolic constraint is either *Ask(c)* (is c true?) and *Tell(c)* (c is true!) where c a literal. Symbolic constraints may exclude activities from being executed. A plan is consistent if it satisfies both temporal and symbolic constraints. TPNs do not specify more than one temporal constraint on the same edge. Consistency is checked by visiting the nodes of the graph from start to end taking one edge (modeling a decision) at a time. If the resulting STN-projection is inconsistent, the algorithm backtracks to the last decision node that still has unexplored outgoing edges. STNDs label nodes and consistency is checked in a hybrid way.

A *Controllable Conditional Temporal Problem (CCTP)* [28] is an optimization problem for temporal plans with choices and thus incomparable with STNDs.

Pike [22] is an executive for temporal plans with both controllable and uncontrollable choices modeled as *Temporal Plan Networks with Uncertainty (TPNUs)*, which extend TPNs with uncontrollable choices. Pike adapts its controllable choices to the uncontrollable ones made by a human. STNDs do not have uncontrollable choices. *CCTPs with Uncertainty (CCTPUs)* [27] address temporal plans with controllable choices and uncontrollable durations, whereas in [18], TPNUs are extended to support uncontrollable durations (strong controllability only). In both works, relaxation techniques are used to restore controllability of an uncontrollable plan. STNDs do not have uncontrollable parts.

Several extensions of *STNs* address uncertain domains. For example, *Simple Temporal Networks with Uncertainty (STNUs)*, [24] add uncontrollable (but bounded) durations by means of a finite set of contingent links, whereas *Conditional Simple Temporal Networks (CSTNs)*, [17], and the *Conditional Temporal Problem (CTP)*, [26] considered formerly, extend STNs by turning the constraints conditional with uncontrollable truth value assignments observable upon the execution of some special kind of time points called observations. Finally, *Conditional Simple Temporal Networks with Uncertainty (CSTNUs)*, [16, 15] merge STNUs and CSTNs, whereas *Conditional Simple Temporal Networks with Uncertainty and Decisions (CSTNUDs)*, [29, 33] adds conditional constraints with controllable truth value assignments decidable upon the execution of some special kind of time points called decision. CSTNUDs



■ **Figure 5** A hierarchy of simple temporal networks. Acronyms containing **D** (resp., **C**) mean that formalisms deal with controllable (resp., uncontrollable) conditionals, whereas those containing **U**, mean that formalisms deal with uncontrollable durations. We highlight uncontrollable parts in red.

encompass all previous formalisms. All these networks extend STNs by adding at least an uncontrollable part. In this work, we do not address any uncontrollable part. STNDs derive from [4] by removing uncontrollable conditionals and from [29, 33] by removing uncontrollable conditionals and uncontrollable durations. This work extends [4] by providing STND-HSCC2 both to speed up the consistency checking phase and to allow for the synthesis of all consistent scenarios. This work is, however, incomparable with [29, 33] as that work employs timed game automata. Figure 5 provides a hierarchy of simple temporal networks.

There also have been attempts to consider time and resources together, e.g., *Access Controlled Temporal Networks* (ACTNs, [6]) and *Conditional Simple Temporal Networks with Uncertainty and Resources* (CSTNURs, [7]), which were preceded by an initial proposal in [8]. However, neither ACTNs nor CSTNURs employ decision time points. Research on temporal networks has inspired a recent line of work in which controllability analysis focused on resource allocation under uncertainty employing a qualitative temporal approach instead of a quantitative one. This is the case of access controlled workflows investigated in [31, 36] and of extensions of constraint networks proposed in [34, 35], where *Constraint Networks Under Conditional Uncertainty* (CNCUs) are introduced (see also [32]). As we said, temporal relations are only qualitative (specifically, “before/after”) and these proposals do not employ decision time points. A short summary of temporal and resource controllability based on constraint networks and considered either in isolation or simultaneously can be found in [30].

Planning as satisfiability was formally introduced in [19, 20] and relies on a set of axioms where any model corresponds to a valid plan. Before that, planning was based on deduction. Recently, more performant SAT encodings have been provided (e.g., [14]). However, none of these approaches is incremental and thus they are incomparable with ours.

Gocht and Balyo [13] provide an incremental SAT solving approach for SAT-based planning and prove that incremental SAT solving outperforms the non-incremental one but they do not address temporal constraints. Our work does not model “transitions” but applies shortest path algorithms, incrementally, on STN-projections.

Temporal induction is an incremental technique to check safety properties on finite state machines and it is strongly related to bounded model checking [12]. It is similar to SAT-based planning and allows for the detection of the unreachability of a goal. Our analysis is not bounded with respect to the “depth”.

Satisfiability modulo theory (SMT, [2]) can describe STNDs by using a fragment of Linear Real Arithmetic called Difference Logic. However, SMT-solvers do not guarantee to find early schedules. A run of an HSCC algorithm and a run of an SMT-solver are not guaranteed to return the same consistent scenarios (Boolean part). A fairer comparison is when both HSCC-algorithms and SMT-solvers seek all consistent scenarios, but then we should make

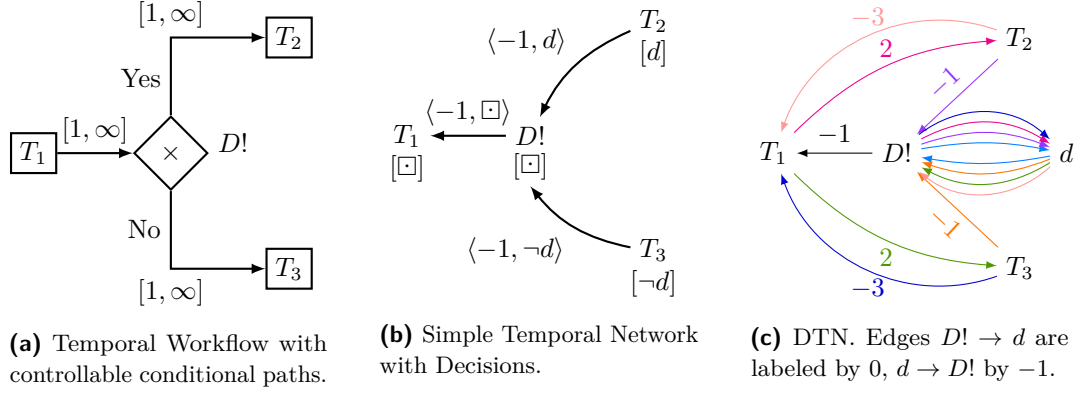


Figure 6 Possible encodings of a temporal workflow (a) into STNDs (b) and into DTNs (c). Note that (c) is the DTN equivalent to (b) streamlined. That is, (b) without labels on time points and plus the constraints $(T_2 - T_1 \leq 2, d)$, $(T_1 - T_2 \leq -3, -d)$, $(T_3 - T_1 \leq 2, -d)$ and $(T_1 - T_3 \leq -3, d)$. There, we chose 2 as an horizon (see [5] for more details on how to streamline a temporal network).

sure that the SMT-solver does not return more than one schedule for each consistent scenario.

Incremental task planning adopts incremental features of SMT-solvers to extend a constraint-based task planning to motion domains [10]. Our approach is not probabilistic and does not consider a motion domain.

7 Conclusions and Future Work

We provided STND-HSCC2, a novel hybrid SAT-based consistency checking algorithm for STNDs. This new version of the algorithm still relies on a SAT solver but differently from the previous one, it exploits partial truth value assignments to hunt down negative cycles in STN-projections as early as possible. The previous algorithm tested STN-projections for negative cycles by iterating on *complete* models returned by the SAT solver. When the SAT solver makes an assumption, we project the STND over the current truth value assignment of the propositions (i.e., partial model) extended with this new assumption. If the projected STN is inconsistent, we add a clause to the SAT solver to exclude that scenario, else we let the solver go. We implemented our approach and provided KAPPA, a tool for STNDs that implements STND-HSCC1 and STND-HSCC2 both supporting the synthesis of single or all consistent scenarios² and we compared the results. The more inconsistent STN-projections an STND admits, the better STND-HSCC2 performs. The solutions saved to file allow for an offline planning in which all decisions are made before starting and all time points have already been scheduled to execute as soon as possible. Finally, we proved that STNDs and DTNs are equivalent. Considering this equivalence result, one could fairly wonder why use STNDs instead of DTNs. Here is a possible reason: STNDs offer a more “structured” language, which, exploiting labels, allows for an easier modeling of temporal workflows with controllable conditional paths (see Figure 6 for a comparison of workflow modeling methods).

As future work, we plan to investigate optimizations to reduce the size of the CNF clauses added on the fly. We also plan to give a metric suggesting the best algorithm to use depending on the form of the STND in input.

² Another minor contribution is the extension of STND-HSCC1 in our tool to support the synthesis of all consistent scenarios.

509 — References

- 510 1 Tomas Balyo, Armin Biere, Markus Iser, and Carsten Sinz. SAT race 2015. *Artificial*
511 *Intelligence*, 241:45–65, 2016.
- 512 2 Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability
513 modulo theories. In *Handbook of Satisfiability*, pages 825–885. IOS Press, 2009.
- 514 3 Daniel Le Berre and Anne Parrain. The sat4j library, release 2.2. *JSAT*, 7(2-3):59–6, 2010.
515 URL: http://jsat.ewi.tudelft.nl/content/volume7/JSAT7_4_LeBerre.pdf.
- 516 4 Massimo Cairo, Carlo Combi, Carlo Comin, Luke Hunsberger, Roberto Posenato, Romeo Rizzi,
517 and Matteo Zavatterì. Incorporating decision nodes into conditional simple temporal networks.
518 In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*,
519 volume 90 of *LIPIcs*, pages 9:1–9:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
520 doi:10.4230/LIPIcs.TIME.2017.9.
- 521 5 Massimo Cairo, Luke Hunsberger, Roberto Posenato, and Romeo Rizzi. A streamlined
522 model of conditional simple temporal networks - semantics and equivalence results. In *24th*
523 *International Symposium on Temporal Representation and Reasoning, TIME 2017*, volume 90
524 of *LIPIcs*, pages 10:1–10:19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. doi:
525 10.4230/LIPIcs.TIME.2017.10.
- 526 6 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatterì. Access controlled
527 temporal networks. In *9th International Conference on Agents and Artificial Intelligence*
528 *(ICAART 2017)*, pages 118–131. INSTICC, ScitePress, 2017. doi:10.5220/0006185701180131.
- 529 7 Carlo Combi, Roberto Posenato, Luca Viganò, and Matteo Zavatterì. Conditional simple
530 temporal networks with uncertainty and resources. *Journal of Artificial Intelligence Research*,
531 64:931–985, 2019. doi:10.1613/jair.1.11453.
- 532 8 Carlo Combi, Luca Viganò, and Matteo Zavatterì. Security constraints in temporal role-based
533 access-controlled workflows. In *6th ACM Conference on Data and Application Security and*
534 *Privacy, CODASPY '16*, pages 207–218. ACM, 2016. doi:10.1145/2857705.2857716.
- 535 9 Patrick R. Conrad and Brian C. Williams. Drake: An efficient executive for temporal plans
536 with choice. *Journal of Artificial Intelligence Research*, 42(1):607–659, 2011.
- 537 10 Neil T. Dantam, Zachary K. Kingston, Swarat Chaudhuri, and Lydia E. Kavradi. Incremental
538 task and motion planning: A constraint-based approach. In *Robotics: Science and Systems*
539 *XII*, 2016. URL: <http://www.roboticsproceedings.org/rss12/>.
- 540 11 Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*,
541 49(1-3):61–95, 1991.
- 542 12 Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. *Electronic*
543 *Notes in Theoretical Computer Science*, 89(4):543–560, 2003.
- 544 13 Stephan Gocht and Tomas Balyo. Accelerating SAT based planning with incremental SAT
545 solving. In *ICAPS 2017*, pages 135–139. AAAI Press, 2017.
- 546 14 Ruoyun Huang, Yixin Chen, and Weixiong Zhang. A novel transition based encoding scheme
547 for planning as satisfiability. In *AAAI 2010*. AAAI Press, 2010.
- 548 15 Luke Hunsberger and Roberto Posenato. Sound-and-Complete Algorithms for Checking the
549 Dynamic Controllability of Conditional Simple Temporal Networks with Uncertainty. In *25th*
550 *International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume
551 120 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17. Schloss
552 Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.TIME.2018.14.
- 553 16 Luke Hunsberger, Roberto Posenato, and Carlo Combi. The Dynamic Controllability of
554 Conditional STNs with Uncertainty. In *Workshop on Planning and Plan Execution for Real-*
555 *World Systems (PlanEx) at ICAPS-2012*, pages 1–8, 2012. URL: [http://arxiv.org/abs/](http://arxiv.org/abs/1212.2005)
556 [1212.2005](http://arxiv.org/abs/1212.2005).
- 557 17 Luke Hunsberger, Roberto Posenato, and Carlo Combi. A Sound-and-Complete Propagation-
558 based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal
559 Networks. In *22nd International Symposium on Temporal Representation and Reasoning*
560 *(TIME 2015)*, pages 4–18. IEEE, 2015. doi:10.1109/TIME.2015.26.

- 561 18 Erez Karpas, Steven James Levine, Peng Yu, and Brian Charles Williams. Robust execution
562 of plans for human-robot teams. In *ICAPS 2015*, pages 342–346. AAAI Press, 2015.
- 563 19 Henry A. Kautz, David A. McAllester, and Bart Selman. Encoding Plans in Propositional
564 Logic. In *KR '96*, pages 374–384. Morgan Kaufmann, 1996.
- 565 20 Henry A. Kautz and Bart Selman. Planning as satisfiability. In *ECAI '92*, pages 359–363.
566 John Wiley & Sons, Inc., 1992.
- 567 21 Phil Kim, Brian C. Williams, and Mark Abramson. Executing reactive, model-based programs
568 through graph-based temporal planning. In *IJCAI 2001*, pages 487–493. Morgan Kaufmann,
569 2001.
- 570 22 Steven James Levine and Brian Charles Williams. Concurrent plan recognition and execution
571 for human-robot teams. In *ICAPS 2014*. AAAI Press, 2014.
- 572 23 Hui Li and Brian Williams. Generalized conflict learning for hybrid discrete/linear optimization.
573 In *CP 2005*, pages 415–429. Springer, 2005.
- 574 24 Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal
575 uncertainty. In *17th International Joint Conference on Artificial Intelligence, IJCAI 2001*,
576 pages 494–502. Morgan Kaufmann, 2001.
- 577 25 Kostas Stergiou and Manolis Koubarakis. Backtracking algorithms for disjunctions of temporal
578 constraints. *Artificial Intelligence*, 120(1):81–117, 2000.
- 579 26 Ioannis Tsamardinos, Thierry Vidal, and Martha E. Pollack. CTP: A new constraint-based
580 formalism for conditional, temporal planning. *Constraints*, 8(4):365–388, 2003.
- 581 27 Peng Yu, Cheng Fang, and Brian Charles Williams. Resolving uncontrollable conditional
582 temporal problems using continuous relaxations. In *ICAPS 2014*. AAAI Press, 2014.
- 583 28 Peng Yu and Brian Charles Williams. Continuously relaxing over-constrained conditional
584 temporal problems through generalized conflict learning and resolution. In *IJCAI 2013*, pages
585 2429–2436. IJCAI/AAAI, 2013.
- 586 29 Matteo Zavatteri. Conditional simple temporal networks with uncertainty and decisions.
587 In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*,
588 volume 90 of *LIPIcs*, pages 23:1–23:17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik,
589 2017. doi:10.4230/LIPIcs.TIME.2017.23.
- 590 30 Matteo Zavatteri. Temporal and resource controllability of workflows under uncertainty.
591 In *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM*
592 *2019*, volume 2420, pages 9–14. CEUR-WS.org, 2019. URL: [http://ceur-ws.org/Vol-2420/](http://ceur-ws.org/Vol-2420/paperDA3.pdf)
593 [paperDA3.pdf](http://ceur-ws.org/Vol-2420/paperDA3.pdf).
- 594 31 Matteo Zavatteri, Carlo Combi, Roberto Posenato, and Luca Viganò. Weak, strong and
595 dynamic controllability of access-controlled workflows under conditional uncertainty. In
596 *Business Process Management - 15th International Conference, BPM 2017*, pages 235–251.
597 Springer, 2017. doi:10.1007/978-3-319-65000-5_14.
- 598 32 Matteo Zavatteri, Carlo Combi, and Luca Viganò. Resource controllability of workflows under
599 conditional uncertainty. In *Business Process Management Workshops (AI4BPM 2019) (to*
600 *appear)*. Springer International Publishing, 2019.
- 601 33 Matteo Zavatteri and Luca Viganò. Conditional simple temporal networks with uncertainty and
602 decisions. *Theoretical Computer Science*, (In press), 2018. doi:10.1016/j.tcs.2018.09.023.
- 603 34 Matteo Zavatteri and Luca Viganò. Constraint networks under conditional uncertainty. In
604 *10th International Conference on Agents and Artificial Intelligence — Volume 2 (ICAART*
605 *2018)*, pages 41–52. SciTePress, 2018. doi:10.5220/0006553400410052.
- 606 35 Matteo Zavatteri and Luca Viganò. Conditional uncertainty in constraint networks. In *Agents*
607 *and Artificial Intelligence*, pages 130–160. Springer, 2019. doi:10.1007/978-3-030-05453-3_
608 7.
- 609 36 Matteo Zavatteri and Luca Viganò. Last man standing: Static, decremental and dynamic
610 resiliency via controller synthesis. *Journal of Computer Security*, 27(3):343–373, 2019. doi:
611 10.3233/JCS-181244.